Topics for the Developers Breakout Session at the LAMMPS User Workshop 2015

New physics features

- linked rigid bonds for TraPPE force field (see p-LINCS in Gromacs)
- dummy atoms (e.g. to support SHAKE with linear molecules, see Gromacs)
- polarizable point dipoles (cf. Tangney/Scandolo JCP paper)
- generalized Born implicit solvent model (GBIS, GBSA) and beyond

Input/output, pre-/post-processing features

- HDF5 file support for trajectory (merge code from Pierre de Beuyl, http://nongnu.org/h5md/)
- native readers for .psf (CHARMM/NAMD) or .parmtop (Amber) or .tpr (Gromacs) or …
- consolidate converter tools to interface to different codes (which ones?) into one package
- VMD GUI plugin frontend for running simple calculations (cf. NAMD GUI plugin in VMD)
- Revisit MPI integration for Python wrapper. what are desirable improvements to wrapper

Improved efficiency and accuracy

- use structs for storage of parameters instead of individual arrays, especially for non-bonded
- general SIMD support via compiler vectorization including: SIMD friendly storage of x-,y-,z- data, SIMD friendly neighborlist (cf. paper by Páll / Hess in CPC), mixed precision SIMD force kernels, vectorizable inline math for potentials using exp()/log()/pow() (vdt++, fastermath)
- improved load-balancing, e.g. bias the result of fix balance using feedback from timer class
- status of KOKKOS vs. GPU vs. Intel and beyond. how to make it easy to use and to contribute to

Build system

- adopt cmake or automake; stop copying sources, but include via variables and vpath
- full support for building multiple targets including libraries from one source tree
- better support for developers using IDEs (eclipse, netbeans, visual c++, …)
- support dynamically loadable styles through a plugin system (incremental updates for binary distributions, commercial add-ons, executable only contains executable code that is used)

Documentation, training and outreach

- reorganize documentation into: 1) a user's guide to introduce using LAMMPS (driven by use cases), 2) an installation guide (with recommendations for deployment on clusters, compiler optimization), 3) a reference guide (documentation of commands/features), 4) a developer's guide
- bundle the above with introductions to relevant theory and publish as a collaborative book
- LAMMPS web page: what should be added, what can be improved, what can be removed?
- improvements to developer's guide, inline documentation using doxygen and sphinx
- coding style guide; recommendations for portability and efficiency
- organize LAMMPS specific developer training (e.g. combined with HPC programming)
- mailing list status alternate ways of user support, especially help for beginners in MD and LAMMPS

Project management and software engineering

- how to make it more attractive for people to contribute to LAMMPS
- status of use of C++11 (and beyond) features; same for STL, Boost.
- how to reduce the growing redundancy and keep the project maintainable
- removal of obsolescent/outdated code? which?
- consolidation of common tasks (e.g. argument parsing)
- public automated regression/integration testing; encourage using bug reporting / issue tracking tool
- organize developer hack-a-thon to work on topics of general concern
- refactoring of LAMMPS following the S.O.L.I.D principles
- move from subversion to git?

Licensing, contracting, and relation with commercial “providers”

- what are the opinions on using GPL for LAMMPS? should (can?) LAMMPS switch to LGPL
- how to set up a way where people can “contract out” a feature they need?
- should there be a “LAMMPS foundation” to be more flexible to take care of the LAMMPS community